



Department Of Computer Science and Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Experiment Name: LRU Page Replacement

Submitted To:

Atia Rahman Orthi

Lecturer,

Department Of Computer Science & Engineering

Submitted By:

Jannatun Saumoon

ID: 21201066

Sec: B2

Problem Statement:

Input:

Page Reference = [7 0 1 2 0 3 0 4 2 3]

Frame Size = 4

LRU (Least Recently Used) page replacement is a memory management algorithm that removes the page that has not been used for the longest time when a new page needs to be loaded and memory is full.

Output: Total Page faults = 6

Solve Process Steps:

Inputs:

Page Reference = [7 0 1 2 0 3 0 4 2 3]


Frame Size = 4

Step	Page	Frames Before	Page Fault	Explanation
1	7	[7]	Yes	Frame empty → insert 7
2	0	[7,0]	Yes	Insert 0
3	1	[7, 0, 1]	Yes	Insert 1
4	2	[7, 0, 1, 2]	Yes	Insert 2
5	0	[7, 1, 2, 0]	No	0 already in frame → move to recent
6	3	[1, 2, 0, 3]	Yes	7 is LRU → remove → insert 3
7	0	[1, 2, 3, 0]	No	0 already in frame → move to recent
8	4	[2, 3, 0, 4]	Yes	1 is LRU → remove → insert 4
9	2	[3, 0, 4, 2]	No	2 already in frame → move to recent
10	3	[0, 4, 2, 3]	No	3 already in frame → move to recent

Output:

Total Page faults = 6

Source Code:

```
 def lru(pages, frame_size):  
    frame = []  
    faults = 0  
  
    for page in pages:  
        if page not in frame:  
            if len(frame) == frame_size:  
                frame.pop(0) # Remove LRU page  
            frame.append(page)  
            faults += 1  
        else:  
            frame.remove(page)  
            frame.append(page)  
  
    print("Total Page Faults:", faults)  
  
# Example usage  
lru([7, 0, 1, 2, 0, 3, 0, 4, 2, 3], 4)
```

 Total Page Faults: 6

Code Steps:

1. Start with an empty frame and set faults to 0.
2. For each page:
 - If not in frame → remove least recently used (if full), add page, count fault.
 - If in frame → move it to the end (most recently used).
1. Print total faults at the end.

Discussion:

1. LRU tracks page usage to always replace the least recently used one.
2. It gives better performance than FIFO by reducing unnecessary replacements.
3. The list structure in the code acts like a usage tracker—most recently used pages are at the end.
4. A bit more complex than FIFO, but it's more intelligent for real-world access patterns.

Conclusion:

LRU is a smarter page replacement algorithm that reduces faults by evicting the least recently used page. Though slightly more complex than FIFO, it offers better performance in most real scenarios where recent access patterns matter.